Autonomous Guidance Navigation and Control for Agile Quadrotors Using Polynomial Trajectory Planning and L_1 Adaptive Control

Mattia Landolfi*†, Saptarshi Bandyopadhyay‡, Jean-Pierre de la Croix§ and Amir Rahmani¶

***Abstract*—We address the challenge to allow efficient autonomous flight in real world environments, both indoor and outdoor. We use a straight-line SE-SCP to find an initial route through the environment and minimum snap trajectory generation using piecewise polynomials. Then, we implement an adaptive robust control able to address some robustness issues for quadrotors in outdoor flight, such as mass variation and wind disturbances. Coupling these techniques we allow high-speed and aggressive autonomous flight through obstacle-dense indoor environments, as well as address outdoor disturbances.**

1. Introduction

Last decade has seen a growing interest in micro aerial vehicles due to their large and increasing range of applications. Recent developments in quadrotors enabled autonomous flight in constrained indoor environments [1]–[5]. Other researches focused on outdoor scenarios [6]–[8]. Quadrotors in outdoor flight require ideal weather conditions. Some works focused on rejecting external constant forces [9] and disturbances in indoor [10] and outdoor [11] spaces. Similar problems have been addressed using disturbance observer [10], [12]. However, there is not yet an approach able to allow efficient high speed autonomous flight in constrained indoor spaces and simultaneously to address outdoor robustness issues. Recent research in motion planning algorithms have succeeded in enabling autonomous quadcopters to fly at high speeds using their full dynamic capabilities [4], [5]. Simultaneously, latest advances in control techniques have allowed fast and robust adaptation to unknown time varing parameters and disturbances [13], and to unknown nonlinearities [14]. These abilities motivate the challenge we address in this paper, which is to allow efficient autonomous flight through cluttered indoor environments and challenging outdoor spaces.

We first use a straight line SE-SCP algorithm [15] to find a waypoint-path through cluttered environments, ignoring the dynamic of the quadrotor. Then, a series of polynomial segments is jointly optimized to link the waypoints into a smooth trajectory that minimize both snap and time allocated to each segment. In order to follow these paths and simultaneously to address wind disturbances and mass variation we present an L_1 based adaptive control for a differentially flat model of the quadrotor.

Main contribution of this paper are: to develop a novel version of an L_1 adaptive control; to couple L_1 adaptive control and polynomial trajectory generation techniques; to improve latest polynomial trajectory generation methods in both computational run time and cost of the final trajectory; to present an application of the L_1 adaptive control technique for a differential flatness system, which is missing in the literature.

2. Physical Model

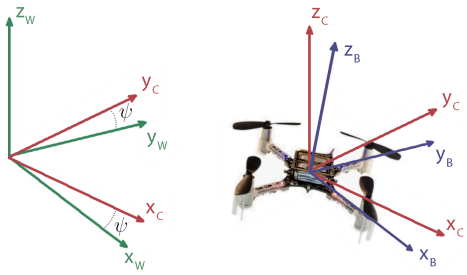

Figure 1. Reference frames

The coordinate systems are shown in Fig 1. We use ${}^{W}R_B = {}^{W}R_C\,{}^{C}R_B$, with components x_B, y_B and z_B, to define the rotation matrix from the body frame, $\mathcal{B}$, to the world frame, $\mathcal{W}$. The rotation matrix ${}^{W}R_C$ denotes the yaw rotation to the intermediate frame $\mathcal{C}$, while ${}^{C}R_B$ represents the pitch and roll effects. We can write the control input as:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T \tag{1}$$

where u_1 is the net body force and, u_2, u_3 and u_4 are the body moments. Force and moments depend on the angular speed that each rotor produces. Dynamic equations describing the acceleration of the center of mass are:

$$m\ddot{\mathbf{r}} = -mgz_W + u_1 z_B - \frac{1}{2} C A_c \rho v_{\mathbf{r}w} |v_{\mathbf{r}w}| \tag{2}$$

where m is mass, g is gravity, $z_W = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and $\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ denotes the position vector of the center of mass in the world frame. While C is the drag coefficients vector, $A_c \in \mathbb{R}^3$ represents the cross-sectional area and $v_{\mathbf{r}w} = \begin{bmatrix} v_{xw} & v_{yw} & v_{zw} \end{bmatrix}^T$ the velocities in x, y, and z directions with respect to the wind. We can rewrite (2) as:

$$\ddot{\mathbf{r}} = -gz_W + \frac{u_1 z_B}{m} + \beta^T f + \delta \tag{3}$$

* *Graduate Visiting Researcher, Jet Propulsion Laboratory (JPL), California Institute of Technology (Caltech).*
† *Graduate Student, Department of Control and Computer Engineering, Politecnico di Torino; mattia.landolfi@studenti.polito.it*
‡ *Postdoctoral Researcher, JPL, Caltech; Saptarshi.Bandyopadhyay@jpl.nasa.gov*
§ *Robotics Systems Engineer, JPL, Caltech; Jean-Pierre.de.la.Croix@jpl.nasa.gov*
¶ *Research Scientist, JPL, Caltech; Amir.Rahmani@jpl.nasa.gov*

where the term $\beta^T f + \delta$ is to model the wind effect. Its parameters are $\delta \in \mathbb{R}^3$, $\beta = \begin{bmatrix}\beta_x & \beta_y & \beta_z\end{bmatrix}$ with $\beta_i \in \mathbb{R}^2$, and $f = \begin{bmatrix}\dot{\mathbf{r}} & \dot{\mathbf{r}} \cdot \dot{\mathbf{r}}\end{bmatrix}^T$. The angular velocity of the body frame in the world frame is denoted by:

$$\omega_{\mathcal{BW}} = p x_B + q y_B + r z_B \tag{4}$$

From Euler equations we determine the angular acceleration:

$$\dot{\omega}_{\mathcal{BW}} = \mathcal{I}^{-1}\left[-\omega_{\mathcal{BW}} \times \mathcal{I}\omega_{\mathcal{BW}} + \begin{bmatrix}u_2 & u_3 & u_4\end{bmatrix}^T\right] \tag{5}$$

with $\mathcal{I}$ as the moment of inertia matrix of the center of mass. The evolution of the rotation matrix ${}^W R_B$ is given by:

$${}^W\dot{R}_B = {}^W R_B \, \hat{\omega}_{\mathcal{BW}} \tag{6}$$

We define the state of the system as:

$$\mathrm{x} = \begin{bmatrix}x & y & z & \dot{x} & \dot{y} & \dot{z} & x_B^T & y_B^T & z_B^T & p & q & r\end{bmatrix}^T \tag{7}$$

Differential flatness [16] of the model without wind dynamics is demonstrated in [2]. Differential flatness for this model can be demonstrated following the same procedure. This property allow us to express all states and inputs in terms of four flat outputs and a finite number of their derivatives. Our choice of flat outputs are the yaw angle and the coordinates of the center of mass in the $\mathcal{W}$ frame:

$$\sigma = \begin{bmatrix}x & y & z & \psi\end{bmatrix}^T \tag{8}$$

A trajectory is defined as a smooth curve in the space of σ:

$$\sigma(t) : [0, \tau] \rightarrow \mathbb{R}^3 \times SO(2) \tag{9}$$

3. Polynomial Trajectory Generation

We define a flat output variable σ_i as a polynomial $P(t)$ of degree N between two points in the flat outputs space. We need to select the coefficients p_n of the polynomial such that its endpoints and their derivatives match desired values at $t = 0$ and $t = \tau$. Furthermore, we want to optimize the following cost function of the derivatives of the polynomial:

$$J = \int_{t=0}^{t=\tau} \left\| \frac{\mathrm{d}^4 \mathbf{r}}{\mathrm{d}t^4} \right\|^2 + \left(\frac{\mathrm{d}^2 \psi}{\mathrm{d}t^2} \right)^2 dt \tag{10}$$

This cost function is used in [2] and effectively discourages abrupt changes in the motor commands to the quadrotor. We can write the complete optimization program as:

$$\begin{aligned} \min_{p} \quad & p^T Q p \\ \text{s.t.} \quad & Ap - b = 0 \end{aligned} \tag{11}$$

where $p^T Q p$ is the cost function rewritten in a quadratic form, with $p \in \mathbb{R}^{N+1}$ as the vector of polynomial coefficients and Q a cost matrix representing our desired penalty on the 4^{th} and 2^{nd} polynomial derivative for position and yaw angle, respectively. The constraint matrix A and vector b can be represented as:

$$A = \begin{bmatrix}A_0 \\ A_\tau\end{bmatrix}, \; b = \begin{bmatrix}b_0 \\ b_\tau\end{bmatrix} \tag{12}$$

where matrices A_0 and A_τ map the coefficients p_n to the polynomial derivatives at the starting and end points, respectively. While the b_0 and b_τ vectors specify the values of the constrained derivatives, including the 0^{th} derivative. Notice that in (11) each flat output variable σ_i is decoupled in both constraints and cost function, hence this minimization problem can be divided into four optimization problems.

Piecewise Polynomial Joint Optimization

Trajectory of a flat output variable is composed by a sequence of segments. We will generate piecewise polynomial trajectories starting from a series of waypoints in the flat outputs space selected by a kinematic planning algorithm. Bry and Richter [4] use a straight-line RRT* [17] to select these waypoints. Our approach is to use a straight-line SE-SCP [15]. This algorithm uses a spherical-expansion-based sampling algorithm to explore the workspace and sequential convex programming techniques to generates a locally optimal trajectory. The SE-SCP algorithm outperforms the RRT* algorithm in terms of computational run time and cost of the final trajectory [15]. Each polynomial segment represents the trajectory between two consecutive waypoints. We need joint optimization to have each pair of consequent polynomial segments to agree on the value of the trajectory derivatives at the in-common waypoint. We formulate the optimization problem over the vector containing all the coefficients of all the polynomial segments. The cost matrix is defined as a block diagonal matrix made of individual Q_k matrices and the constraints are composed by a combination of two sets. The first set enforce derivatives continuity at each joint between segments:

$$\begin{bmatrix}-A_\tau^i & A_0^{i+1}\end{bmatrix} \begin{bmatrix}p_i \\ p_{i+1}\end{bmatrix} = 0 \tag{13}$$

where p_i is the vector containing all the coefficients of the i^{th} polynomial segment. The second set of constraints is to specify desired derivative values:

$$\begin{bmatrix} A_0^0 & 0 & \dots & 0 \\ 0 & A_0^1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_0^K \\ 0 & 0 & \dots & A_\tau^K \end{bmatrix} \begin{bmatrix}p_0 \\ p_1 \\ \vdots \\ p_K\end{bmatrix} = \begin{bmatrix} b_0^0 \\ b_0^1 \\ \vdots \\ b_0^K \\ b_\tau^K \end{bmatrix} \tag{14}$$

where vector b_0^i specifies derivatives at the beginning of the i^{th} polynomial segment while vector b_τ^K specifies derivatives at the end point of the last segment. A combination of the rows of (13) and (14) will enable us to enforce continuity as well as specify the set of desired derivative values.

This constrained optimization procedure works well for short trajectories, since to joint optimize several segments we need to increase the degree of the polynomials with a consequent increasing in complexity of the optimization program that can become ill-conditioned. In order to optimize long-range path requiring many waypoints and segments can be used an unconstrained reformulation of the optimization problem where the endpoint derivatives substitute the polynomial coefficients as decision variables.

Unconstrained Optimization

In order to reformulate to an unconstrained optimization we first substitute the i^{th} individual segment constraint equations $p_i = A_i^{-1} b_i$ into the cost function. Then, we re-arrange the decision terms b_i such that specified/fixed derivatives are grouped together (b_F) as well as the unspecified/free derivatives (b_P):

$$J = \begin{bmatrix}b_F \\ b_P\end{bmatrix}^T C A^{-T} Q A^{-1} C^T \begin{bmatrix}b_F \\ b_P\end{bmatrix} \tag{15}$$

with

$$CA^{-T}QA^{-1}C^T \triangleq R = \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix} \tag{16}$$

where A and Q are block-diagonal matrices of A_i and Q_i from the i^{th} segment, while C is a permutation matrix. Matrix C can be seen as multiplication of two matrices, each one with a specific role:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_1 \\ \vdots \\ b_{k-1} \\ b_{k-1} \\ b_k \end{bmatrix} = C_1^T \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{k-1} \\ b_k \end{bmatrix} = C_1^T C_2^T \begin{bmatrix} b_F \\ b_P \end{bmatrix} \tag{17}$$

Differentiating (15) after partitioning and equating to zero we obtain the optimal unspecified/free derivatives vector:

$$b_P^* = -R_{PP}^{-1} R_{FP}^T b_F \tag{18}$$

The coefficients of the polynomials can be recovered back-mapping derivatives into the space of coefficients using the appropriate constraint matrix.

The unconstrained optimization problem must be solved involving dense computation or sparse solver methods. Otherwise, since C is sparse and A^{-1} and Q are sparse block-diagonal, singularities issues arise in the use of the two parts of the R matrix.

Time Allocation

Amount of time τ_i allocated to each segment is required for the construction of the optimization problem. Varying these segment times we can improve the final solution with respect to a cost function. Since the total trajectory time is not known a priori, we let it to change in the optimization in order to trade-off between minimizing the original cost function and total trajectory time. We find optimal segment times using the following cost function:

$$J_\tau = \int_{t=0}^{t=\tau} \left\| \frac{\mathrm{d}^4 \mathbf{r}}{\mathrm{d}t^4} \right\|^2 dt + c_\tau \sum_{i=0}^{K} \tau_i \tag{19}$$

where c_τ is a user-specified penalty on time. This new cost function has a minimum value for some finite $\sum_{i=0}^{K} \tau_i$ which depends on c_τ. We begin with an initial guess of segment times and then we solve the minimization problem using a gradient descent technique where we compute the directional derivative for K vectors denoted by g_i as:

$$\nabla_{g_i} J_\tau = \frac{J_\tau(\tau + h g_i) - J_\tau(\tau)}{h} \tag{20}$$

where h represents some small number, while vectors g_i are constructed such that the i^{th} element has value 1 and all other elements are 0. The algorithm for the gradient descent technique is shown in Algorithm 1. Our choice was

Algorithm 1 Gradient Descent

1: Initialize τ
2: **while** not convergence or max number of iterations **do**
3: $\quad \tau \leftarrow \tau - \gamma \nabla J(\tau)$

to maintain a step size γ fixed at each iteration. Furthermore, we defined γ to be inversely proportional to the penalty on time c_τ. This decision is to prevent the gradient growing exaggeratedly for high penalties on time, as well as to use reasonable γ values in case of small c_τ.

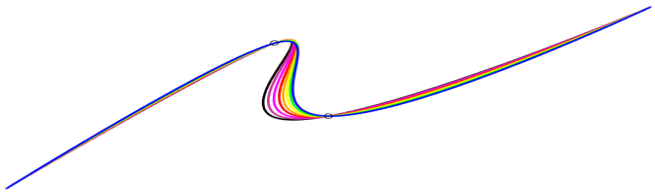

Figure 2. Gradient descent iterations, color-coded by total traversal time. The initial guess of segment time is 10 s for each segment (black), while the final optimized total trajectory time is 6.68 s (blue), 2.25 s, 1.91 s and 2.52 s allocated to each segment respectively.

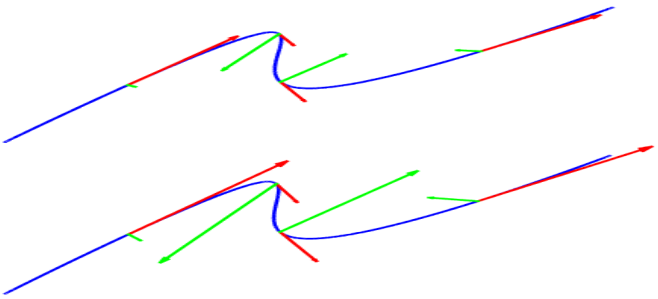

Figure 3. Segment time optimization with c_τ = 1000 (top) and c_τ = 10000 (bottom). Vectors for acceleration (green) and velocity (red) are shown. The optimal total trajectory times are 9.83 s and 6.68 s respectively.

Fig 2 shows the iterative refinement of segment times through gradient descent in which the time allocated to each segment is decreased from an initial guess (black) to an optimal value (blue). The segment times ratio also shifts to minimize the cost function in (19). Fig 3 shows optimized trajectories from a given arrangement of waypoints adopting two different time penalties. Vectors for acceleration (green) and velocity (red) are smaller in the top trajectory due to the lower c_τ. Regardless of the value of c_τ the geometric shape of the final optimized trajectory does not vary, showing that the optimal ratios of segment times and c_τ are independent for the same set of waypoints.

Solving Practical Issues

If after optimization we detect an intersection between a trajectory segment and an obstacle, we re-optimize the polynomial (including time allocation) adding an extra waypoint halfway between the two ends of that segment, as in [3]. This additional waypoint is collision-free because located on the optimal piecewise-linear path generated by the straight-line SE-SCP. This re-optimization procedure is repeated recursively until we obtain a collision-free trajectory. Usually the addition of one or two midpoints in a given segment resolves collisions in indoor environments [4]. Several strategies could be used to reduce the need of re-optimization, like placing an elevated cost on paths that go near obstacles or slightly enlarging obstacles during the route-finding phase.

Another issue involves input constraints. We would like to have that no portion of the commanded trajectory requires

control inputs that exceed what actuators are capable of providing. We manage this issue during the time allocation phase. Knowing that the optimal segment times ratio is independent of the total trajectory time we firstly optimize the ratio of times through gradient descent technique neglecting actuator constraints. Then, we scale the total trajectory time in a separate univariate optimization, maintaining the optimal times ratio constant, until an actuator limit is reached or the modified cost function is minimized.

Time Computation

Time computation of a similar polynomial trajectory generation technique, as well as comparison with RRT* using polynomial steer function, are presented in [4]. However, main differences in our approach are: a different time allocation cost function, which does not include the yaw contribution of the original cost function in (10), resulting in reduced time computation without degenerating correctness of the final solution; the use of SE-SCP algorithm instead of RRT* for the waypoints selection through the environment. Using SE-SCP we improve performances in both computational run time and cost of the final trajectory. Comparison between the two algorithms is presented in [15].

4. L_1 Based Adaptive Control

In this section, we will implement an L_1 based adaptive controller for the presented differential flatness system to deal with wind disturbances as well as time-varying mass. In this controller, only the parasitic drag are considered as disturbances, while other drag effects are ignored.

L_1 Adaptive Control Structure

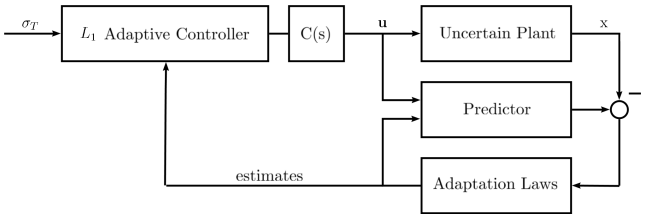

Figure 4. L_1 Adaptive Control Structure

Figure 4 shows the closed-loop system. The low-pass filter $C(s)$, with gain $C(0) = 1$, ensures that the estimation loop is decoupled from the control loop [13]. Using dynamic equations in (3) we can write the uncertain plant system as:

$$\ddot{\mathbf{r}} = -gz_W + \alpha u_1 z_B + \beta^T f + \delta \tag{21}$$

where α is the inverse of the time-varying mass. We choose the state predictor with the same structure of the plant:

$$\hat{\ddot{\mathbf{r}}} = -gz_W + \hat{\alpha} u_1 z_B + \hat{\beta}^T \hat{f} + \hat{\delta} \tag{22}$$

where the adaptive estimates $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\delta}$ replace the unknown parameters.

L_1 Controller

The controller has to follow specified trajectories, $\sigma_T(t) = [\mathbf{r}_T(t)^T, \psi_T(t)]^T$. We compute the first control input projecting the desired force vector onto the actual z_B:

$$u_1 = C(s)(\mathbf{F}_{des} \cdot z_B) \tag{23}$$

where desired force vector is computed as follows:

$$\mathbf{F}_{des} = \frac{\ddot{\mathbf{r}}_T - \hat{\beta}^T f - \hat{\delta} + g z_W}{\hat{\alpha}} + \epsilon \tag{24}$$

where ϵ is used to correct numerical errors due to the physical model formulation. In particular, the projection of $\mathbf{F}_{des}$ onto the $\mathcal{B}$ frame z axis (23) and the re-projection onto the z_B axis in the quadrotor physical model (21), lead to small position errors. These impact on the z axis more than on the other axes. Since we model the predictor as the quadrotor physical model, some effects of these errors, although diminished, remain hidden and accumulate over time. Since a PD controller is able to manage these imprecision acting directly on the position error, we decide to model factor ϵ as a PD controller:

$$\epsilon = -K_p e_p - K_v e_v \tag{25}$$

where e_p and e_v are the errors on position and velocity, while K_p and K_v are positive definite gain matrices. Next we compute the desired rotation matrix R_{des}. Desired z_B is computed observing that it is along the desired thrust vector:

$$z_{B,des} = \frac{\mathbf{F}_{des}}{\|\mathbf{F}_{des}\|} \tag{26}$$

We then construct unit vector $x_{C,des}$ using specified yaw angle $\psi_T(t)$ and we compute $x_{B,des}$ and $y_{B,des}$ as follows:

$$x_{C,des} = \begin{bmatrix}\cos\psi_T & \sin\psi_T & 0\end{bmatrix}^T \tag{27}$$

$$y_{B,des} = \frac{z_{B,des} \times x_{C,des}}{\|z_{B,des} \times x_{C,des}\|} \tag{28}$$

$$x_{B,des} = y_{B,des} \times z_{B,des} \tag{29}$$

provided that we never have $z_{B,des}$ parallel to $x_{C,des}$, we can uniquely determine the desired rotation matrix R_{des}. We can fix the problem of singularity observing that $-x_{B,des}$ and $-y_{B,des}$ are also consistent with ψ_T and $z_{B,des}$ and therefore directly check which solution is closer to the actual quadcopter orientation. Next we compute the body frame components of desired angular velocity, $\omega_{\mathcal{BW},des}$, as:

$$h_\omega = \frac{1}{\hat{\alpha} \cdot \mathbf{F}_{des} \cdot z_B}(\dot{\mathbf{a}}_T - (z_{B,des} \cdot \dot{\mathbf{a}}_T) z_{B,des}) \tag{30}$$

$$p_{des} = -h_\omega \cdot y_{B,des}, \quad q_{des} = h_\omega \cdot x_{B,des} \tag{31}$$

$$r_{des} = \omega_{\mathcal{CW}} \cdot z_{B,des} = \dot{\psi}_T z_W \cdot z_{B,des} \tag{32}$$

where h_ω is the projection of $\frac{1}{\hat{\alpha} \cdot \mathbf{F}_{des} \cdot z_B} \dot{\mathbf{a}}_T$ onto the $x_B - y_B$ plane. Now we compute the three remaining control inputs:

$$\begin{bmatrix}u_2 & u_3 & u_4\end{bmatrix}^T = -K_R e_R - K_\omega e_\omega \tag{33}$$

where K_R and K_ω are diagonal gain matrices, while e_R and e_ω define error on orientation and angular velocity:

$$e_R = \frac{1}{2}(R_{des}^T {}^W R_B - {}^W R_B^T R_{des})^\vee \tag{34}$$

$$e_\omega = {}^B[\omega_{\mathcal{BW}}] - {}^B[\omega_{\mathcal{BW},des}] \tag{35}$$

where $^\vee$ denotes the vee operator. Finally we compute the rotor speeds to achieve the desired control input simply inverting the appropriate linearization.

The estimates $\hat{\alpha}(t)$, $\hat{\beta}(t)$ and $\hat{\delta(t)}$ are governed by the following adaptation laws:

$$\dot{\hat{\alpha}}(t) = -k_\alpha u_1(t) \tilde{\ddot{\mathbf{z}}}(t), \quad \hat{\alpha}(0) = \hat{\alpha}_0 \tag{36}$$

$$\dot{\hat{\beta}}_{\mathbf{i}}(t) = -k_{\beta\mathbf{i}} f_{\mathbf{i}}(t) \tilde{\ddot{\mathbf{i}}}(t), \quad \hat{\beta}_{\mathbf{i}}(0) = \hat{\beta}_{\mathbf{i}0}, \quad \mathbf{i} \in \{\mathbf{x}\ \mathbf{y}\ \mathbf{z}\} \tag{37}$$

Table 1. RUNNING TIME ON MATLAB

Controller	Runtime (ms)
PD	0.10
L_1	0.12

$$\dot{\hat{\delta}}(t) = -k_\delta \tilde{\ddot{\mathbf{r}}}(t), \quad \hat{\delta}(0) = \hat{\delta}_0 \tag{38}$$

where $k_\alpha \in \mathbb{R}^+$, $k_\delta \in \mathbb{R}^3$ and $k_{\beta \mathbf{i}} = diag(\rho_1,\ \rho_2)$ are the adaptation gains, and $\tilde{\ddot{\mathbf{r}}}$ represents the acceleration error:

$$\tilde{\ddot{\mathbf{r}}}(t) = \hat{\ddot{\mathbf{r}}}(t) - \ddot{\mathbf{r}}(t) = \begin{bmatrix} \tilde{\hat{\mathbf{x}}}(t) & \tilde{\hat{\mathbf{y}}}(t) & \tilde{\hat{\mathbf{z}}}(t) \end{bmatrix}^T \tag{39}$$

Another way to compute the adaptive estimates could be involving the use of the projector operator, as in [13] and [14]. The projection operator ensures that the adaptive estimates $\hat{\alpha}(t)$, $\hat{\beta}(t)$ and $\hat{\delta}(t)$ remain inside the defined compact sets $[\alpha_l,\ \alpha_u]$, $[-\beta_b,\ \beta_b]$ and $[-\delta_b,\ \delta_b]$, defined in [14].

Remarks

Proofs of stability and convergence for a pure L_1 adaptive controller are presented in [13] and [14], while comparison with several adaptive controllers can be found in [18]. The factor ϵ, other than correct numerical errors due to the physical model formulation, has effects on stability and convergence of the controller, and on parameters estimation. Due to the nature of ϵ, analysis on stability and convergence for the nonlinear PD controller developed in [2] for the presented quadrotor model are also valid for this L_1 based controller. Moreover, the ϵ term cause errors in parameters estimation. However, these errors do not reduce performances since the estimated parameters change in order to have a predictor system response as close as possible to that of the system. Nevertheless, in the case we would like to have perfectly match between disturbances and estimated parameters ϵ gains must be set to 0 and a forgetting factor has to be added to the wind estimation to put steady-state errors back to zero in case of no disturbances. These steady-state errors are due to two behaviors: on the z axis the effect of the mass variation and the wind is almost the same; when orientations different from hovering, disturbances on the z axis generate disturbances also on the other axes, involving activation of the estimation parameters of x and y axes.

5. Implementation and Results

The following will be a comparison between the presented L_1 controller and the nonlinear PD controller developed in [2]. We generated 9^{th} degree polynomial (10 coefficients) trajectories and we set second and third derivatives to 0 at the beginning and end points of the trajectory. We implemented the simulations in Matlab and Simulink environments under the following: in order to have simulations as close as possible to the practical environment we discretized our controllers at $100Hz$; we decided to express the attitude error as in (34) though it has no physical meaning; we used real data of a Crazyflie 2.0 nano quadcopter [20]; we took into account rotors limitations. A discretization frequency of $100Hz$ allows our controllers to run safely. Table 1 shows running time on matlab for both our controllers. The running time of the L_1 is comparable with that of the PD.

In the L_1 implementation, high gains in the ϵ factor

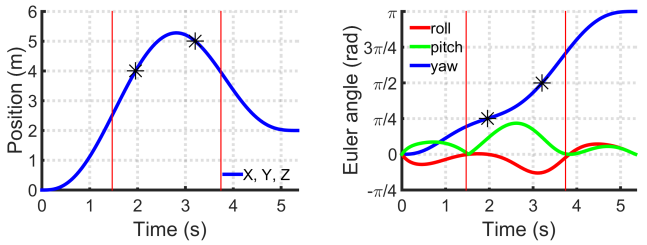

Figure 5. Sample 3-segment trajectory. Position in x,y and z, and yaw (blue) were generated using the polynomial trajectory generation technique, while roll (red) and pitch (green) were extrapolated from the differentially flat model.

can lead ϵ to take over the L_1 in the behavior against disturbances. We noticed that, in the simulation without disturbances, one or two order of magnitude lower for K_p and K_v than those used for the PD controller allows good manage of the numerical issues due to the model without degenerating the L_1 typical control response. However, in practice can be used lower values for ϵ with respect to those used for the PD controller. For K_R and K_ω, instead, we set the same values as those used in the PD implementation.

The discretization of the controller put an upper limit to the adaptation gains. The analysis of these bounds is not straight forward. We decided to use a tuning strategy based on the analysis of the upper limits obtained considering the mass variation and wind disturbances separately, as in [11]. We first set k_δ and then the other adaptive gains as:

$$k_\alpha = \frac{k_\delta}{{u_{1,\max}}^2}, \quad \rho_1 = \frac{k_\delta}{{\dot{\mathbf{r}}_{\max}}^2}, \quad \rho_2 = \frac{k_\delta}{{\dot{\mathbf{r}}_{\max}}^4} \tag{40}$$

where $u_{1,\max}$ and $\dot{\mathbf{r}}_{\max}$ can be extrapolated from the generated polynomial trajectory and eventually adjusted for the task the quadrotor has to perform.

We tested controllers on different trajectories and both yield good and comparable tracking performances. We then tested the behavior response against mass variation. Fig 6 shows errors and control output comparison when a mass of $15 \cdot 10^{-3}\, Kg$ (56% of the quadcopter weight) is added to the quadcopter at the first red vertical line of the sample 3-segment trajectory in Fig 5 and then released at the second red vertical line. We can notice that the response of the L_1 is faster in both cases. In particular, the overshoot in the position error is lower and the error decrease faster to lower values. Another emergent property of the L_1 is in the controller outputs. As we can see in Fig 6, the response is smoother with respect to that generated by the PD controller. In practice, oscillations in the input signal of the rotors are always not appreciated because of the motors dynamics.

Finally we tested our controllers against wind disturbances, Fig 7. We decided to simulate wind disturbances as a step input in β and δ parameters at the first red vertical line of the sample trajectory in Fig 5. However, for simulation results closer to real flights would be better to use the Dryden model introduced in [19], which estimates wind disturbances as a filtered white-noise.

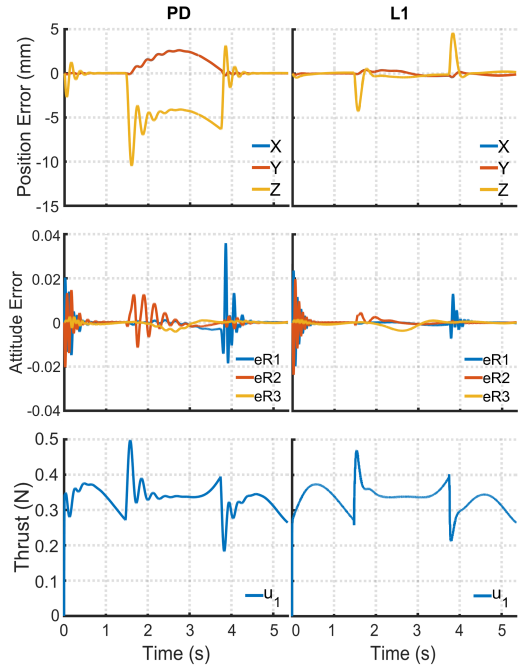

Figure 6. Adding and releasing a mass for PD (left) and L_1 (right)

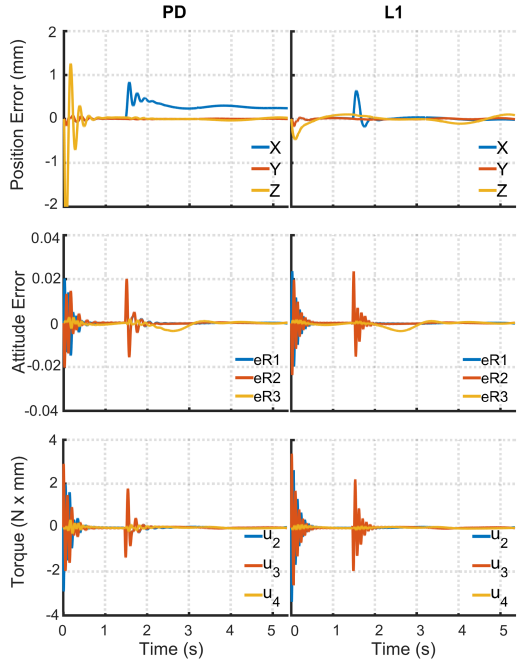

Figure 7. Response to wind on the x axis for PD (left) and L_1 (right)

6. Conclusion

We presented an L_1 based adaptive controller to control both position and attitude, and simultaneously, to address wind disturbances and mass variation. The key advantages of this L_1 controller over the PD controller are: fast adaptation against disturbances resulting in rapid decrease in the position error; lower overshoot error caused by disturbances; smoother control input. We also presented a detailed design of minimum snap trajectory generation using piecewise polynomials and time allocation method that trades off between snap and time minimization. The polynomial trajectory generation procedure is computationally fast to be used for real time purpose. Coupling these techniques we enabled efficient autonomous flight through both obstacle-dense indoor and challenging outdoor environments.

Acknowledgment This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] A. Boeuf, J. Cortés, R. Alami and T. Siméon, *"Planning agile motions for quadrotors in constrained environments"*, IEEE International Conference on Intelligent Robots and Systems (IROS 2014), 2014.

[2] D. Mellinger and V. Kumar, *"Minimum snap trajectory generation and control for quadrotors"*, IEEE international conference on robotics and automation (ICRA'11), 2011.

[3] J. Pan, L. Zhang and D. Manocha, *"Collision-free and smooth trajectory computation in cluttered environments"*, International Journal of Robotics Research 31: 11551175, 2012.

[4] A. Bry, C. Richter, A. Bachrach and N. Roy, *"Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments"*, The International Journal of Robotics Research 1-34, 2015.

[5] C. Richter, A. Bry and N. Roy, *"Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments"*, International Symposium of Robotics Research (ISRR 2013), 2013.

[6] L. V. Santana, A. S. Brandão and M. Sarcinelli-Filho, *"Outdoor Waypoint Navigation with the AR.Drone Quadrotor"*, International Conference on Unmanned Aircraft Systems (ICUAS), 2015.

[7] A. Symington, R. De Nardi, S. Julier and S. Hailes, *"Simulating Quadrotor UAVs in Outdoor Scenarios"*, IEEE International Conference on Intelligent Robots and Systems (IROS 2014), 2014.

[8] P. Moonumca, Y. Yamamoto and N. Depaiwa, *"Adaptive PID for Controlling a Quadrotor in a Virtual Outdoor Scenario: Simulation Study"*, Proc. of IEEE International Conference on Mechatronics and Automation, 2013.

[9] D. Cabecinhas, R. Cunha and C. Silvestre, *"A Nonlinear Quadrotor Trajectory Tracking Controller With Disturbance Rejection"*, American Control Conference (ACC), 2014.

[10] H. Wang and M. Chen, *"Trajectory tracking control for an indoor quadrotor UAV based on the disturbance observer"*, Transactions of the Institute of Measurement and Control 1-18, 2015.

[11] M. Q. Huynh, W. Zhao and L. Xie, *"L_1 adaptive control for quadcopter: Design and implementation"*, 13th International Conference on Control Automation Robotics & Vision (ICARCV), 2014.

[12] A. Modirrousta and M. Khodabandeh, *"Adaptive Robust Sliding Mode Controller Design for Full Control of Quadrotor with External Disturbances"*, International Conference on Robotics and Mechatronics, 2014.

[13] C. Cao and N. Hovakimyan, *"L_1 Adaptive Controller for Systems with Unknown Time-varying Parameters and Disturbances in the Presence of Non-zero Trajectory Initialization Error"*, International Journal of Control, vol. 81, no. 7, pp. 1147-1161, July, 2008.

[14] C. Cao and N. Hovakimyan, *"L_1 Adaptive Controller for a Class of Systems with Unknown Nonlinearities"*, American Control Conference, 2008.

[15] F. Baldini, S. Bandyopadhyay, R. Foust, S. J. Chung, A. Rahmani, J. P. de la Croix, A. Bacula, C. M. Chilan, F. Y. Hadaegh, *"Fast Motion Planning for Agile Space Systems with Multiple Obstacles"*, AIAA/AAS Astrodynamics Specialist Conference, 2016.

[16] M. J. Van Nieuwstadt and R. M. Murray, *"Real-time trajectory generation for differentially flat systems"*, International Journal of Robust and Nonlinear Control, vol. 8, pp. 995-1020, 1998.

[17] S. Karaman and E. Frazzoli, *"Sampling-based algorithms for optimal motion planning"*, The International Journal of Robotics Research 30: 846-894, 2011.

[18] J. D. Bošković and N. Knoebel, *"A Comparison Study of Several Adaptive Control Strategies for Resilient Flight Control"*, AIAA Guidance, Navigation, and Control Conference, 2009.

[19] H. W. Liepmann, *"On the Application of Statistical Concepts to the Buffeting Problem"*, Journal of the Aeronautical Sciences 19(12):793-800, 1952.
[20] "Bitcraze AB", https://www.bitcraze.io